1 OVERVIEW

1.1 What is a Virtual Hub?

An ENERGIC OD Virtual Hub (VH) is a single point of access to harmonized geospatial open data. Geospatial open data are served by many distributed systems and comes in different flavours, with heterogeneous interfaces for discovery and access, and different metadata and data models. Therefore, without a hub, to use geospatial open data, a developer should spend a lot of time searching for useful data and become an expert in many different specifications. Moreover, applications should waste computing power and time for harmonizing the accessed information before being able of using it. A VH addresses these problems connecting heterogeneous data sources and serving geospatial open data through one or more homogeneous interfaces, in a common format, coordinate reference system and resolution. Doing that, a VH greatly facilitates the use of geospatial open data, solving interoperability issues and allowing developers to focus on their application logic.

1.1 What do Virtual Hubs provide?

The VHs provide two main functionalities:

1. *Semantically enhanced discovery of geospatial open data*: the user can ask the VH to provide a list of available datasets filtered in terms of geographic coverage, temporal extent, data provider, content keywords. The VH can semantically interpret the keywords using external knowledge bases. For example, if a keyword is in English and the knowledge base provides its translation in other languages, also the datasets annotated in different languages will be considered during the search process.
2. *Harmonized access to geospatial open data*: the user can ask the VH to provide a set of data identified by an Internet address (a URL, for example provided by a preliminary discovery query). Upon request, data can be also processed on-the-fly for subsetting (cutting data on a specific spatio-temporal volume), reprojection (change of coordinate reference system), resampling (change of resolution) encoding (change of format).

1.2 Finding and accessing geospatial open data

Although all the VHs provide the same functionalities, each VH differs in terms of objectives. A VH may have a national or regional scope providing only open data on a specific area, another VH may have a thematic scope providing open data useful for a specific application domain. This affects which data are available through a specific VH, how the semantic search is oriented, which format and coordinate reference systems are supported and so on.

The VH manager decides about the configuration of the VH. The user can check the characteristics of the VH, and the available datasets through the VH access portal.

1.3 How is a VH implemented?

For those interested, technical information about the implementation of ENERGIC OD VHs is available in the official ENERGIC OD deliverable “Virtual Hubs - System Architecture (Second Release)” [1].

2 GETTING STARTED

2.1 Connecting to the VH

2.1.1 The VH Web Portal

The VH is accessible through a Web Portal at a specific Internet address:

`<VH_portal> := http://myvh.energic-od.eu/` (example)

Pointing a browser to the `<VH_portal>` address an end user or a developer can access information and functionalities, including:

- VH Data Portal
- Developers section

(The graphical aspect of the VH Web Portal may differ between different VHs depending on VH requirements and scope.)

Figure 1 Example of VH Web Portal. The *Search* button gives access to the VH Data Portal.
(The graphical aspect of the VH Data Portal may differ between different VHs depending on VH requirements and scope.)

2.1.2 The VH Data Portal

The VH Data Portal provides discovery and access of geospatial datasets. It has its own Internet address:

`<VH_data_portal> := http://myvh.energic-od.eu/data` (example)

(The graphical aspect of the VH Data Portal may differ between different VHs depending on VH requirements and scope.)

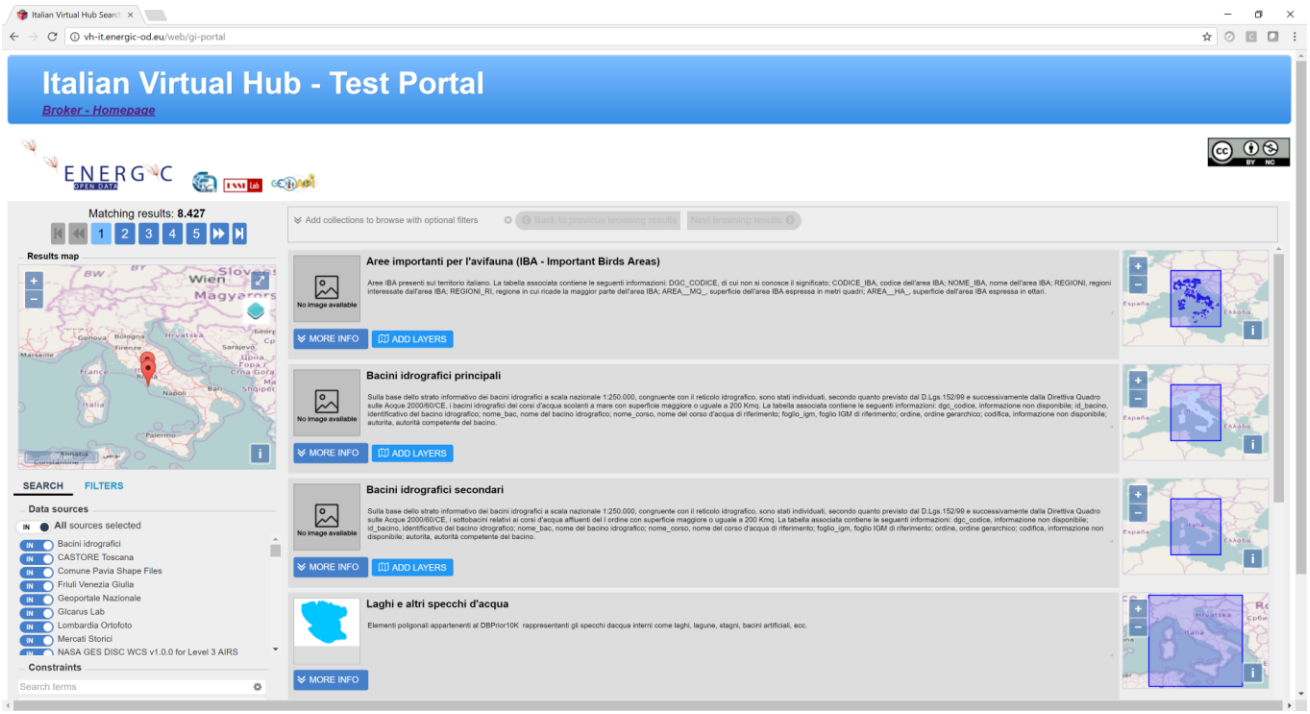

Figure 2 Example of VH Data Portal.
(The graphical aspect of the VH Data Portal may differ between different VHs depending on VH requirements and scope.)

An end-user can access the VH Data Portal for discovering and accessing datasets directly from the Web. A developer can access the VH Data Portal to explore available datasets of interest for his/her planned application.

2.1.3 The VH Information Page

Through the developer's section it is possible to access the VH information page, providing basic information for connecting to the VH. It has its own Internet address:

`<VH_info_page> := http://myvh.energic-od.eu/info` (example)

Pointing a Web browser to the info page, the VH landing page shows up. It provides a set of information and links for VH managers and VH developers.

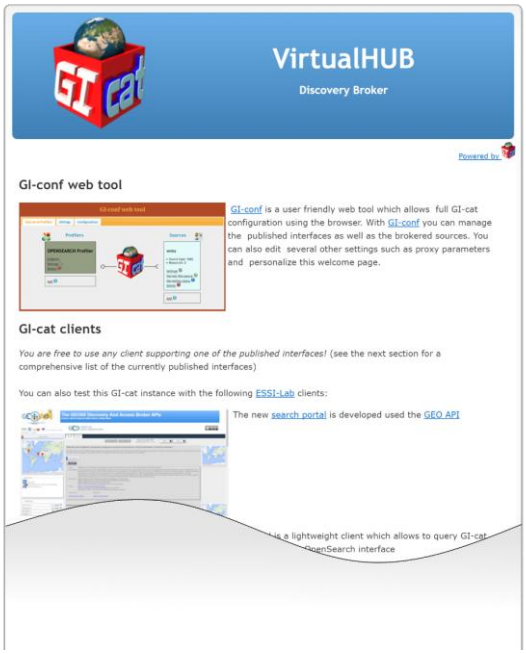

Figure 3 VH Landing page showing useful information and links

2.2 Developing with a VH

By the developer's point-of-view, a VH is a black box which takes care of mediating and harmonizing the information coming from multiple providers on behalf of the user.

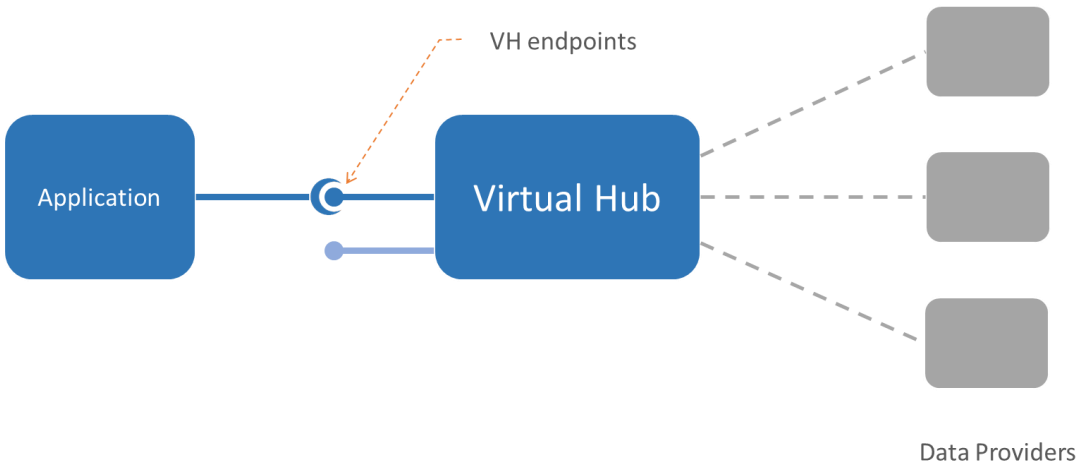

Figure 4 Application communicates with the VH through an Internet endpoint

The communication between an application and the VH happens through one or more of the exposed VH endpoints.

Not all the developers are the same. They have different requirements and different expertise on the use of geospatial data. Some are geospatial data experts and they would like to port applications already working

on existing data systems, others would like to integrate the VH geospatial data in their system, others would like to simply integrate geospatial information in their Web and mobile apps.

For that reason, there are three different ways to interact with the VH. They are not mutually exclusive, and the user can mix them if he/she needs to. However, each way targets a specific kind of developer to address specific requirements.

- *Standard geospatial interfaces*: The VH exposes de-iure and de-facto standard interfaces for geospatial data discovery, access and visualization, such as OGC CSW, WFS, WCS, WMS. They allow to interact with the VH as if it is a single standard compliant data system. This interaction allows porting of existing applications which already uses standard interfaces and it is suitable for developers who are expert of geospatial interoperability who would like to have full control on the interaction with data sources.
- *RESTful API*: The VH exposes the main functionalities for geospatial open data discovery and access through a RESTful API. It allows an easy interaction with the VH through any programming language which supports the HTTP protocol and the JSON message encoding.
- *Web API*: This is a HTML5+Javascript+CSS library which facilitates the development of web and mobile applications, hiding the interaction with the VH behind the behavior of simple Javascript objects.

3 DEVELOPING WITH STANDARD GEOSPATIAL INTERFACES

3.1 Discovery and access

A VH exposes a set of standard geospatial interfaces that an application can access. The specific type of exposed interfaces is defined by the VH manager. A user can check the exposed interfaces through the VH information page. A list of geospatial data discovery interfaces is presented.

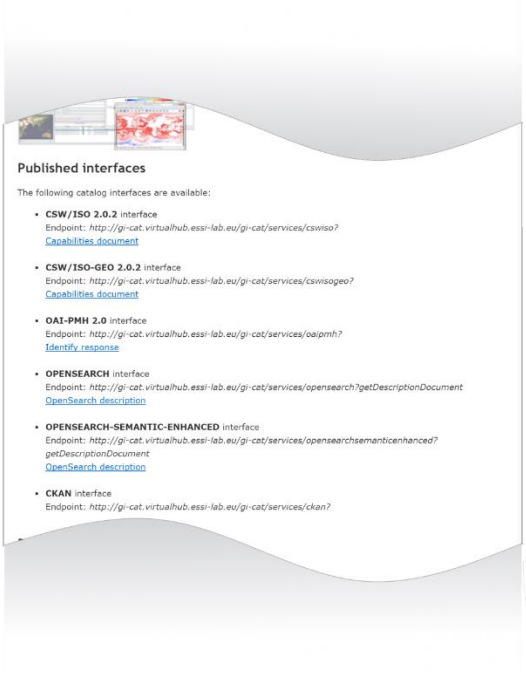

Figure 5 Information on the standard interfaces exposed by the VH with their endpoints

A sample entry is:

CSW/ISO 2.0.2 interface

Endpoint: *http://gi-cat.virtualhub.essi-lab.eu/gi-cat/services/cswiso?*

Capabilities document

It says that the VH exposes a CSW/ISO 2.0.2 interface at the endpoint *http://gi-cat.virtualhub.essi-lab.eu/gi-cat/services/cswiso?*. CSW/ISO 2.0.2 means that the interface is compliant with the version 2.0.2 of the Catalogue Service for Web defined by the Open Geospatial Consortium, and the metadata model is ISO (19115). The documentation is available through the web sites of the standard organizations. In this case, for example, the OGC CSW 2.0.2 specification is accessible at http://www.opengeospatial.org/standards/cat and the ISO 19115 specification is accessible at http://www.iso.org/iso/catalogue_detail.htm?csnumber=53798.

All the discovery interfaces provide a list of datasets available through the VH. The description of each entry provides one or more access services to download the datasets. The list includes the original access services to download the dataset from the data provider, and one or more specific entries to download the dataset through the VH. Using one of these entries it is possible to exploit the VH access capabilities such as the dataset transformation (subsetting, interpolation, resampling and format change) even if the original server is not able to provide such capabilities. Figure 6 shows an example of dataset description as presented by the VH portal. The Links section reports three access links. The first one is the original OGC WMS service exposed by the data provider; the second one is an access service provided by the VH through a proprietary advanced interface (named GI-axe); the third one is a standard compliant OGC WMS service exposed by the VH.

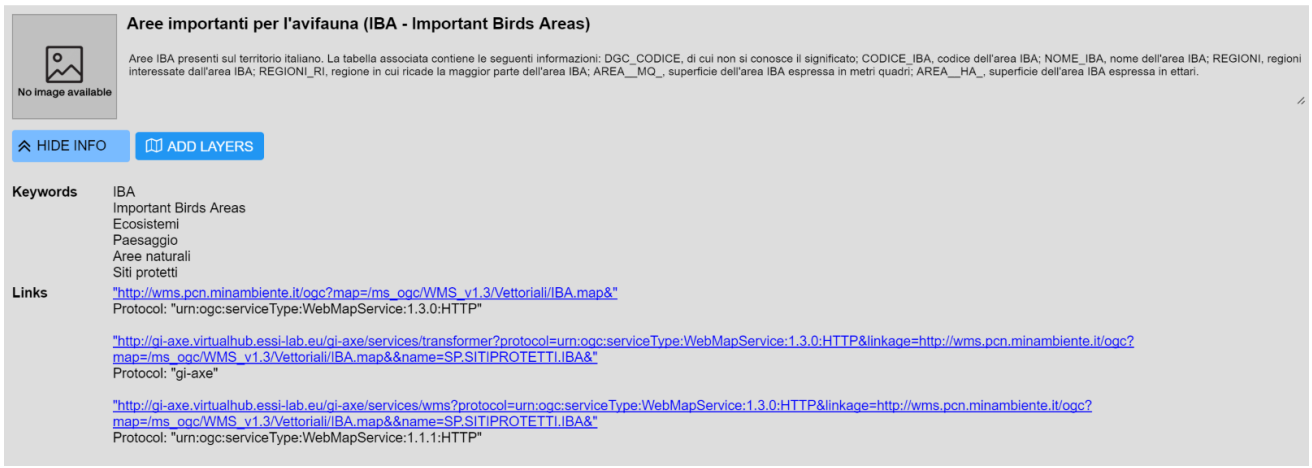

Figure 6 An example of dataset description as presented by the VH Data Portal

A user who is expert in geospatial technologies can develop an application compliant with the specifications supported by the VH. Figure 7 shows an example of an application interacting through two standard interfaces with the VH and with one of the original data sources.

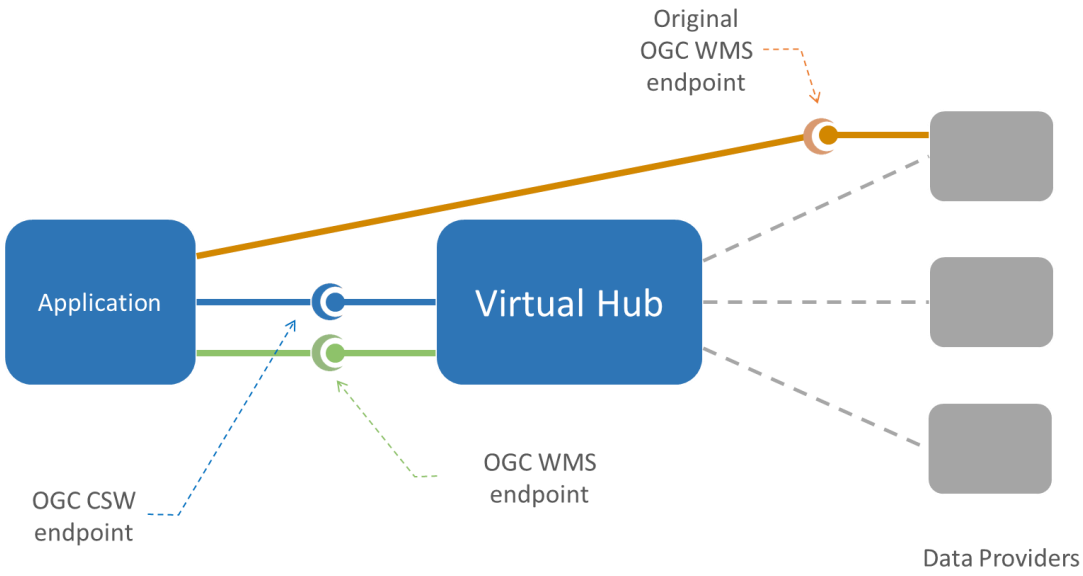

Figure 7 An example of application interacting with the VH through standard interfaces

4 DEVELOPING WITH THE RESTFUL API

4.1 Discovery and access

4.1.1 Introduction

REST (REpresentational State Transfer) is an architectural style for the implementation of resource sharing systems. According to its creator, REST is the original and proper architectural style of the World Wide Web. The World Wide Web is currently used as a general purpose infrastructure to deliver services according to different architectural styles, however, over recent years, there was a great interest in providing services fully compliant with the REST architectural style. Therefore, many Web systems provide access to their functionalities through so-called RESTful APIs. Although the concept of APIs does not make sense from a rigorous REST point of view, the term RESTful API is now improperly used to refer a way to interact with a software system through request and response messages to resources identified by an URI. In RESTful APIs, message content is usually structured in JSON or XML format.

The ENERGIC OD VH follows this trend providing a simple way to interact with the hub by exchanging JSON objects with resources identified by URIs.

Conforming to the REST architectural style, the VH RESTful API gives direct access to resources supporting the Create-Retrieve-Update-Delete (CRUD) pattern. The main resources exposed by the VH RESTful API are:

- *Sources*: the list of data sources connected to the VH instance
- *Datasets*: the list of datasets accessible through the VH instance
- *Dataset*: the single dataset

According to REST constraints, each resource is identified by a URI. In the VH RESTful API, the resource URI includes a VH base endpoint (the same address of the VH Web Portal) and a mnemonic path conventionally defined:

- *Sources* `<VH_base>/sources`
- *Datasets* `<VH_base>/datasets`
- *Dataset* `<VH_base>/datasets/{id}`

Where {id} should be replaced with the specific dataset ID.

Each resource supports on or more CRUD actions implemented as HTTP requests. For example, to retrieve the list of sources, the user must send a HTTP GET request to `<VH_endpoint>/sources`:

```
GET <VH_base>/sources
```

The user then receives back a JSON object including information on the data sources connected to the VH:


```
{
  "resultSet": {
    "size": 7,
    "start": 1,
    "pageSize": 7,
    "pageCount": 1,
    "pageIndex": 1
  },
  "reports": [
    {
      "id": "GBIF",
      "type": "composed",
      "title": "GBIF",
      "harvested": false
    },
    {
      "id": "WEBSRVENCAT",
      "type": "composed",
      "title": "Webservice Energy Catalog",
```

```
      "harvested": true
    },
    {
(…continue…)
```

Figure 8 shows a subset of the ENERGIC OD resource model, with direct interaction with resources made possible through REST interfaces implemented using HTTP verbs.

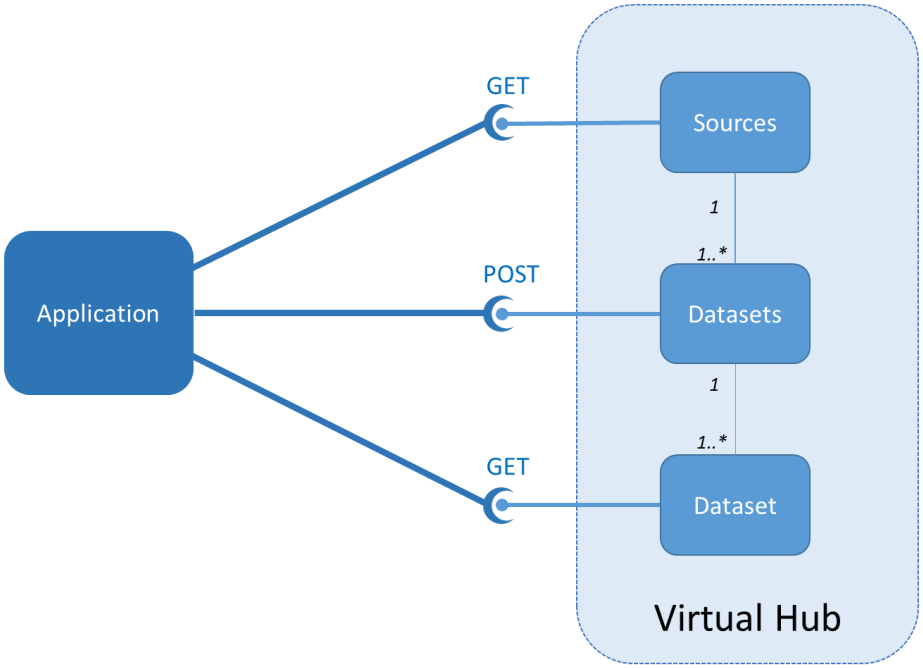

Figure 8 A subset of the VH resource model and related REST interface implemented in HTTP

The full RESTful API documentation available online provides a detailed description of the exposed resources, their URIs, supported actions and message format, including examples and live demos.

4.1.2 RESTful API documentation

(*Annex I reports the official documentation of the RESTful API version 1.0.0-alpha. The latest version of the documentation is available at the persistent URL* http://api.eurogeoss-broker.eu/dab/api-rest-docs/ .)

5 DEVELOPING WITH THE WEB API

5.1 Discovery and access

5.1.1 Introduction

The term Web API is generically used for referring to some implementation of an application programming interface for Web applications. As such, a Web API may be a RESTful API or a library implemented in some web application language. The ENERGIC OD VH uses the term Web API in the latter sense, to identify a Javascript library facilitating the development of web and mobile apps using the capabilities of VHs.

The ENERGIC OD Web API is a Javascript library providing a set of predefined software objects whose properties and behaviour hide the complexity of the interaction with the VH, presenting the developer only significant methods.

Figure 10 shows the main Web API's objects and their behavior in a typical application:

- *DAB*: It is the Web API entry point. This object represents a VH discovery and access broker which is capable to access heterogeneous data sources in a homogeneous way. During the creation phase, it is associated to a specific VH instance. It provides discovery capabilities to find relevant datasets provided by the VH.
- *ResultSet*: An object provided as a result of a DAB discovery operation. It represents a set of datasets matching the DAB discovery constraints.
- *Paginator*: An object associated to the ResultSet which allows to retrieve results as subsets (pages). It helps handling results avoiding long and unmanageable lists.
- *Page*: An object representing a subset of a ResultSet.
- *GINode*: An object representing a geoinformation resource (typically a dataset) which is a single result of the discovery operation and part of a Page.
- *Layer*: An object which is a dataset representation ready to be used by other Javascript libraries and environments like OpenLayers.

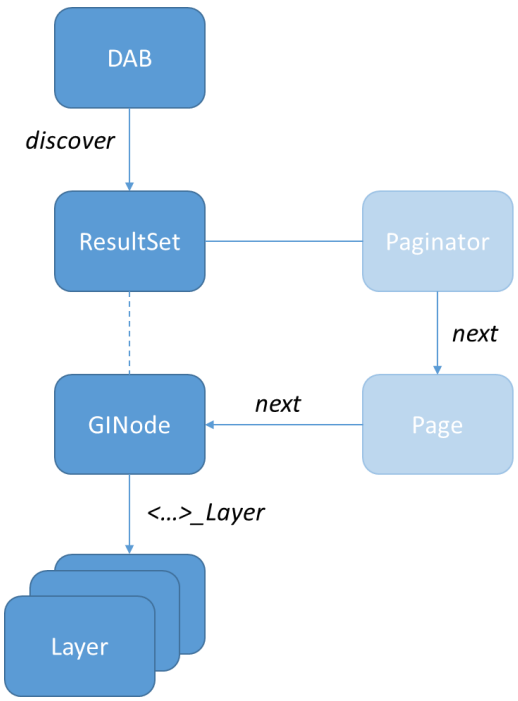

Figure 9 The main concepts represented as objects in the Web API

The *discover* method of the *DAB* object, allows to retrieve a *ResultSet* object, containing all the *GINode* objects satisfying the query constraints which are specified as a JSON object passed to the *discovery* call. The access to the *GINode* objects is mediated by a *Paginator* object linked with the *ResultSet*, which allows to access limited sets of *GINode*s as *Page*s. The *GINode* content can be accessed as a layer of different types.

The full Web API documentation available online provides a detailed description of objects, properties, and methods including examples and live demos.

5.1.2 Web API documentation

(*Annex II reports the official documentation of the DAB JavaScript API version 1.4.2-beta. The latest version of the documentation is available at the persistent URL* http://api.eurogeoss-broker.eu/docs/index.html .)

6 DEVELOPING WITH THE WEB API

6.1 Discovery and access

6.1.1 Introduction

The term Web API is generically used for referring to some implementation of an application programming interface for Web applications. As such, a Web API may be a RESTful API or a library implemented in some web application language. The ENERGIC OD VH uses the term Web API in the latter sense, to identify a Javascript library facilitating the development of web and mobile apps using the capabilities of VHs.

The ENERGIC OD Web API is a Javascript library providing a set of predefined software objects whose properties and behaviour hide the complexity of the interaction with the VH, presenting the developer only significant methods.

Figure 10 shows the main Web API's objects and their behavior in a typical application:

- *DAB*: It is the Web API entry point. This object represents a VH discovery and access broker which is capable to access heterogeneous data sources in a homogeneous way. During the creation phase, it is associated to a specific VH instance. It provides discovery capabilities to find relevant datasets provided by the VH.
- *ResultSet*: An object provided as a result of a DAB discovery operation. It represents a set of datasets matching the DAB discovery constraints.
- *Paginator*: An object associated to the ResultSet which allows to retrieve results as subsets (pages). It helps handling results avoiding long and unmanageable lists.
- *Page*: An object representing a subset of a ResultSet.
- *GINode*: An object representing a geoinformation resource (typically a dataset) which is a single result of the discovery operation and part of a Page.
- *Layer*: An object which is a dataset representation ready to be used by other Javascript libraries and environments like OpenLayers.

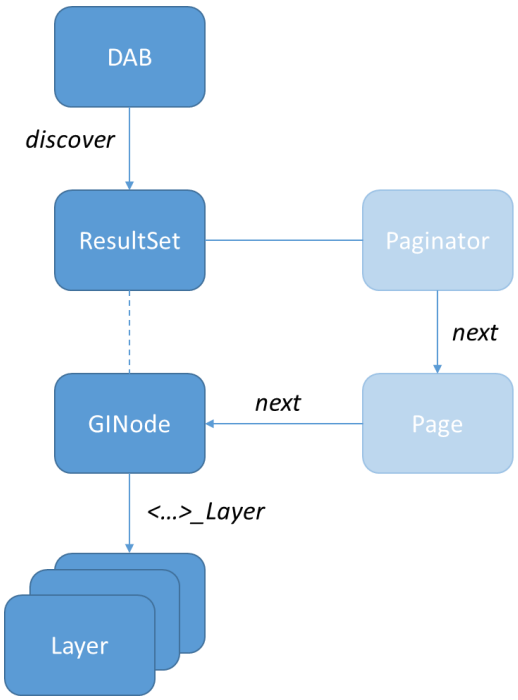

Figure 10 The main concepts represented as objects in the Web API

The *discover* method of the *DAB* object, allows to retrieve a *ResultSet* object, containing all the *GINode* objects satisfying the query constraints which are specified as a JSON object passed to the *discovery* call. The access to the *GINode* objects is mediated by a *Paginator* object linked with the *ResultSet*, which allows to access limited sets of *GINode*s as *Page*s. The *GINode* content can be accessed as a layer of different types.

The full Web API documentation available online provides a detailed description of objects, properties, and methods including examples and live demos.

6.1.2 Web API documentation

(*Annex II reports the official documentation of the DAB JavaScript API version 1.4.2-beta. The latest version of the documentation is available at the persistent URL* http://api.eurogeoss-broker.eu/docs/index.html .)

REFERENCES

[1] ENERGIC OD Consortium, «Virtual Hubs - System Architecture (second release) - D5.1b,» 2016.